Towards realism in drawing areas of interest on architecture diagrams

Byelas, Heorhiy; Telea, Alexandru

Published in:
Journal of visual languages and computing

DOI:
10.1016/j.jvlc.2008.09.001

Publication date:
2009

Journal of Visual Languages and Computing 20 (2009) 110–128

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

Towards realism in drawing areas of interest on architecture diagrams

Heorhiy Byelas, Alexandru Telea *

Department of Mathematics and Computer Science, University of Groningen, Nijenborgh 9, 9747 AG Groningen, The Netherlands

ARTICLE INFO

Article history:
Received 24 October 2007
Received in revised form
4 September 2008
Accepted 12 September 2008

Keywords:
UML diagrams
Software visualization
Information visualization
Empirical evaluations
Visual shape comparison

ABSTRACT

Areas of interest (AOIs) are defined as groups of elements of system architecture diagrams that share some common property. Visualizing AOIs is a useful addition to plain diagrams, such as UML diagrams. Some methods have been proposed to automatically draw AOIs on UML diagrams. However, it is not clear whether actual users perceive the results of such methods to be better or worse as compared to human-drawn AOI, and what needs to be improved. We present here a process of studying and improving the perceived quality of computer-drawn AOI. For this, we conducted a qualitative evaluation that delivered insight in how users perceive the quality of computer-drawn AOIs as compared to hand-drawn diagrams. Following these results, we derived and implemented several improvements to an existing algorithm for computer-drawn AOIs. Next, we designed a distance metric to quantitatively compare different AOI drawings, and used this metric to show that our improved rendering algorithm creates drawings which are closer to (good) human drawings than the original rendering algorithm. We present here the results of the user evaluation, our improved algorithm for drawing AOIs, and the quantitative analysis performed to compare different drawings. The combined user evaluation, algorithmic improvements, and quantitative comparison method support our claim of having improved the perceived quality and understandability of AOI rendered on architecture diagrams.

1. Introduction

UML diagrams are among the methods of choice for system architects to describe and understand software architectures and designs, e.g. the structural and functional relations between the various interfaces, components, or roles [1]. Their design and exploration is supported by many tools, both commercial, such as Rational Rose [2], Borland Together [3], Telelogic Tau [4], and open source, such as ArgoUML [5]. Software elements that share a common property are of particular interest in system analysis, e.g. "all high-reliability components", "all components using over 1 MB of memory", "all components introduced in the system version 2.3", or "all components in the same thread". We call such a set of elements an *area of interest* (AOI). AOIs can be defined using software metrics [6,7] computed by existing analysis tools [8].

AOIs and their underlying defining metrics are usually displayed in tabular format. Recently, a method was proposed for visually combining AOIs and architecture (UML) diagrams [9]. The chief advantage of this method is that it shows AOIs in-place on the diagrams, so it enables users to correlate concerns (AOIs) with system structure (diagrams). At the same time, the method does not alter the diagram layout, which can be used to encode structural system properties. The method in discussion renders AOIs as soft, fuzzy shapes surrounding the diagram elements, by a combination of geometric and

* Corresponding author. Tel.: +31 40 2475008; fax: +31 40 2468508.
E-mail addresses: h.v.byelas@rug.nl (H. Byelas), a.c.telea@rug.nl (A. Telea).

1045-926X/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jvlc.2008.09.001

texture-based techniques (see, e.g. Fig. 22b). Roughly, the rendering algorithm tries to imitate the way humans would draw AOIs in practice using pen and paper. The rendering method scales computationally well to tens of areas and hundreds of elements, and produced results which were found quite appealing by potential users.[1]

However, for the computer method to be actually effective in practice, some important questions still remain to be answered: Do actual users like computer-drawn AOIs comparably to hand-drawn AOIs? If not, why, and how can we improve the computer-drawn AOIs so that they resemble more closely good-quality hand-drawn ones? In this paper, we present our quest to measure and improve the perceived quality of computer-drawn AOI. To evaluate the quality of AOIs, we designed and executed a detailed empirical evaluation. From the evaluation results, we distilled salient strengths and weaknesses of the original AOI-rendering algorithm [9] and of hand-drawn areas. Our conclusion was that hand-drawn areas, although quite variable across different humans, are perceived as easier to understand than computer-drawn ones. Two main drawbacks of computer-drawn areas were found: incorrect exclusion of overlapping elements, and unnatural flow-of-hand. Next, we modified and extended the original AOI algorithm to address these limitations. Finally, we designed a distance metric to compare AOI renderings, and showed that the results of our improved algorithm are closer to (good) human drawings than the results of the original rendering algorithm.

This paper is structured as follows. Section 2 reviews related work in visualizing AOIs on diagrams and evaluating quality aspects related to diagram drawing. Section 3 overviews the original AOI algorithm [9]. Section 4 details the algorithmic limitations of the original method. Section 5 presents the empirical evaluation we conducted to compare the quality of computer and human drawings. Section 6 presents our new technique that improves the rendering of AOIs on diagrams by correct geometric exclusion of elements and natural flow-of-hand rendering. Section 7 presents a quantitative comparison of the human-drawn and computer renderings. Section 8 presents and discusses the results of our evaluation and proposed algorithmic improvements for our goal of enhancing the quality of computer-rendered AOI. Finally, Section 9 concludes the paper.

2. Related work

Visualizing AOI can be described with the 5D model of Marcus et al. [10]: task, audience, target, medium, and representation. Our task is to understand how various system aspects (the AOIs) map on some system description (the UML diagram). Our audience is mainly composed of software architects. Our visualization target is a set of diagrams, together with AOIs specified as sets of diagram elements. The visualization medium is a modified UML diagram viewer [11] that combines rendering diagrams and AOIs. The representation enriches classical UML diagram drawings with AOIs drawn as smooth soft-textured shapes with a new technique.

Although current software design and system modeling methods produce numerous attributes apart from structural data, e.g. metrics related to performance, trust, security, debugging, and deploying, there is little widespread support for showing such attributes on classical (UML) diagrams. Modeling tools, e.g. Rational Rose [2] or Together [3], are standard ways to visualize UML diagrams, but have little support for AOI beyond boxes. The SUMLOW system allows users to construct UML models by converting their hand-drawn sketches to the UML formalism [12]. This system could support hand-drawn AOIs but does not address the issue of automatically constructing such annotations from a specification as they are not part of the UML standard. Drawing AOIs as boxes without changing the base diagram layout yields unacceptably high visual clutter. Tools such as Rigi [13], Prefuse [14], or MetricView [11], often used in reverse engineering and reengineering, show an AOI by marking its elements with icons scaled, colored, and shaped to show metric values. Yet, inferring AOIs from such markers is hard for diagrams with many overlapping AOIs. One can also move all elements in an AOI close to each other and next draw a surrounding frame [15]. However, diagrams are often laid out manually with great care. It is well known that changing the layout every time one changes the AOIs destroys the user's mental map. Moreover, layout adaption does not work when one needs to show several AOIs at the same time.

A separate class of methods draws AOIs as smooth shapes around their respective elements. Methods such as metaballs [16], H-BLOB [17], and 2D implicit surfaces [18] compute AOIs as isosurfaces of some potential function, or distance field, based on the elements' locations. However, it is hard to control both the smoothness and tightness of such isosurfaces. Worse, the isosurface connectivity highly depends on a correct isovalue which cannot be easily chosen automatically [17]. Finally, distance fields and isosurfaces are computationally expensive.

A recent technique in the class of smooth shapes draws AOIs by shrinking and smoothing the convex hull of the enclosed elements [9]. The features targeted by this technique are:

1. AOIs do not change a given diagram layout.
2. AOIs drawing is real-time, even for large diagrams.
3. AOIs correctly surround the enclosed elements.
4. AOIs do not clutter the diagram or each other.
5. Computer- and human-drawn AOIs should resemble each other.

The first two requirements were clearly satisfied by the proposed technique, by construction. The technique did not correctly handle requirement 3, as we shall see in Section 4.1. Moreover, it was not measured how precisely requirements 4 and 5 are satisfied.

A separate body of related work concerns evaluating the quality of a visual depiction of system (UML) architectures. Specifically, we want to assess which type

[1] The proposed method has obtained the "best paper award" at ACM SoftVis'06 [9].

of AOI rendering is the best, and why. Since we are not aware of specific studies to evaluate the quality of AOI renderings, we shall consider the wider range of evaluating quality aspects of UML diagram renderings. Purchase et al. have conducted numerous user experiments to assess the comprehensibility, aesthetics, and user preferences of UML (and similar) diagram renderings [19–22]. Such results are valuable both as methodology and lessons learned, yet they cannot be applied directly to our problem, since AOIs are an extension of the standard UML notation. Several authors propose frameworks and methodologies to evaluate the comprehensibility and overall quality of UML models [23–28]. Still, the question "what are the good quality criteria for visual modeling languages?" is not exhaustively answered.

There is a large body of related work in the area of quality attributes of hand-drawn diagrams and diagram annotations beyond UML. Notably, Plimmer et al. have presented several systems for human annotation of computer-made diagrams [29,30]. In particular, the RCA tool manages user-drawn annotations to fit around edited source code in an IDE, which is similar to our requirement that AOIs should fit the elements they enclose, regardless of their layout [31]. Beautification issues of hand-drawn diagrams and annotations are discussed by Plimmer and Grundy [32] and Yeung et al. [33]. Identified desirable issues such as annotation line smoothness, annotation constrainment to user-specified layouts, and the use of a natural stroke or flow-of-hand, are all directly relevant to our computer-drawn AOIs. Our particular challenge is to generate such annotations entirely automatically, rather than starting from a user sketch.

One emerging conclusion from the previous work is that plain, unannotated UML is often hard to comprehend and can perform better if extended by task-specific annotations. Our AOI are precisely such an annotation, useful to show cross-cutting concerns atop of a given system structure. Since this is a new notation, the characteristics that make for a good AOI drawing have not yet been studied in particular. Our aim is to construct a computer algorithm that renders AOIs similarly to good hand-drawn AOIs. The above-mentioned requirements of the technique in [9] attempt to capture the *in abstracto* quality criteria of a good AOI drawing. Yet, to assess the *perceived* quality of an AOI drawing, we need a specific study. In the remainder of this paper, we present such a study and an improved computer AOI-rendering algorithm based on the study results.

3. AOI construction

Let us first briefly overview the construction of the AOIs as presented in [9]. AOIs are built in two stages. First, a contour is built around the elements, in three steps (Figs. 1b–d). Given the 2D bounding boxes $(b_{1i}, b_{2i}, b_{3i}, b_{4i})$ of the elements e_i in the AOI (Fig. 1a), one first computes the convex hull $C = \{q_i\}$ of the corners $\{b_{ij}\}$ (Fig. 1b). The hull is a start approximation for the AOI shape. The hull is next subsampled (Fig. 1c) so that the distance $\delta = |q_i - q_{i+1}|$ between consecutive points is a given small fraction of the hull perimeter $|C| = \sum_i |q_i - q_{i+1}|$, e.g. $\delta = 0.01|C|$.

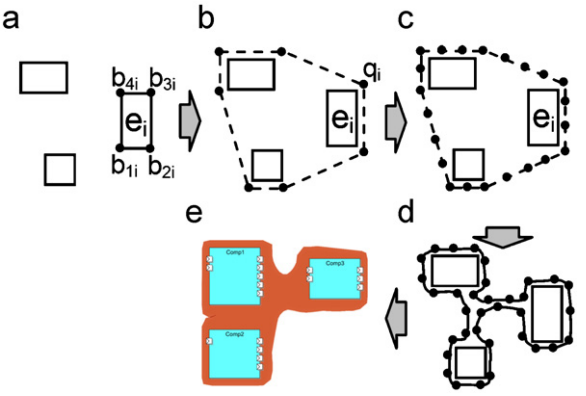

Fig. 1. Area of interest algorithm.

Next, the subsampled contour $\{q_i\}$ is deformed so that it fits tightly the elements inside and also becomes smoother (Fig. 1d). For this, every point q_i is moved to q_i' along the normal $\vec{n}$ to the line segment $(q_{i-1}q_{i+1})$, i.e. $q_i' = q_i + \varepsilon_n \vec{n} + \varepsilon_s(q_{i-1} + q_{i+1})/2$. Here, ε_n and ε_s are parameters that control the shrinking and smoothing strengths, respectively. Good values are $\varepsilon_n = 0.005|C| = 0.5\delta$ and $\varepsilon_s = |q_{i-1} - q_{i+1}|/4$. If $\{q_i\}$ are in counterclockwise order, q_i moves inwards in C. This shrinks the contour, but also moves q_i towards the middle of the segment $(q_{i-1}q_{i+1})$, which is the well-known geometric Laplacian smoothing operation [34]. To prevent contour self-intersection, a point q_i is moved only if

$$d = \min\left(\min_{|j-i|>1} |q_i - q_j|, \min_j |q_i - p_j|\right) > 2\delta \quad (1)$$

i.e. if the contour point q_i is farther from all corners p_j and other contour points q_j (except its immediate neighbors q_{j-1}, q_{j+1}) than 2δ. To prevent the contour to become too densely or sparsely sampled, we check the distances $\min_{|j-i|>1}|q_i - q_j|$ between the moved point q_i and its neighbors, and insert or remove points if these fall outside the range $[0.5\delta, 2\delta]$. Points are moved until N_{max} iterations are done.

In the second stage, given a contour constructed as above, AOIs are drawn in three steps. If we want filled areas, the contour is triangulated and drawn in the area's color, else this step is skipped. Next, the contour points q_i are offset outwards along the contour normal $\vec{n}$, i.e. $q_i' = q_i + \varepsilon_n \vec{n}$, where ε_n is the shrink factor. This creates a narrow band along the contour (Fig. 2a). This band is rendered using a gradient-like OpenGL texture with transparency $T(x, y) = x^k$ (Fig. 2b) mapped on the quadrilaterals $(q_i q_{i+1} q_{i+1}' q_i')$. Here, $T = 0$ yields fully transparent pixels and $T = 1$ fully opaque ones, creating a soft border effect.

Fig. 3 shows several deformation steps for a simple AOI, starting from the convex hull until a tight shape, reached after 20 iterations. Different visual effects can be achieved by tuning the parameters. The texture profile parameter k controls the contour fuzziness: $k<1$ yields "hard", crisp contours, while $k>1$ yields soft, fuzzy ones. A good value is $k = 0.7$, which was used for all images in this paper. The number of iterations N controls the

contour tightness: $N>10$ gives tight contours, whereas $N<5$ gives loose, more rounded ones.

4. Technical limitations

The original AOI-rendering algorithm (Section 3) has a number of limitations of which we were already aware during design and testing. We discuss these next.

4.1. Ineffective exclusion

The contour constructed as described in Section 3 may erroneously overlap, or *include*, elements which are logically not in the AOI. To mark such elements as outside the AOI, the original method draws a rectangular *eraser texture* around the element, creating a narrow soft white border, which should suggest the exclusion of that element. The texture transparency profile, shown in Fig. 4, is given by the following function:

$$f(x) = \begin{cases} 1, & x<b \\ \left(\frac{x-b}{b}\right)^k, & x \geqslant b \end{cases} \quad (2)$$

Here, b controls the eraser's white border width.

Consider Fig. 5 where elements A–D are in the area and E, F are outside. The eraser method works reasonably well if we draw *filled* areas and the overlapping elements are

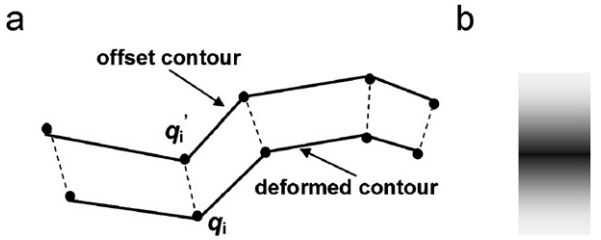

Fig. 2. Drawing soft borders with textures. (a) Band construction and (b) band texture.

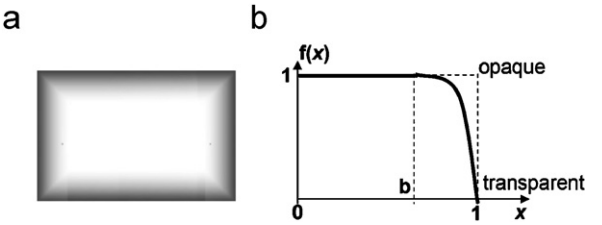

Fig. 4. Eraser texture design. (a) Texture and (b) profile.

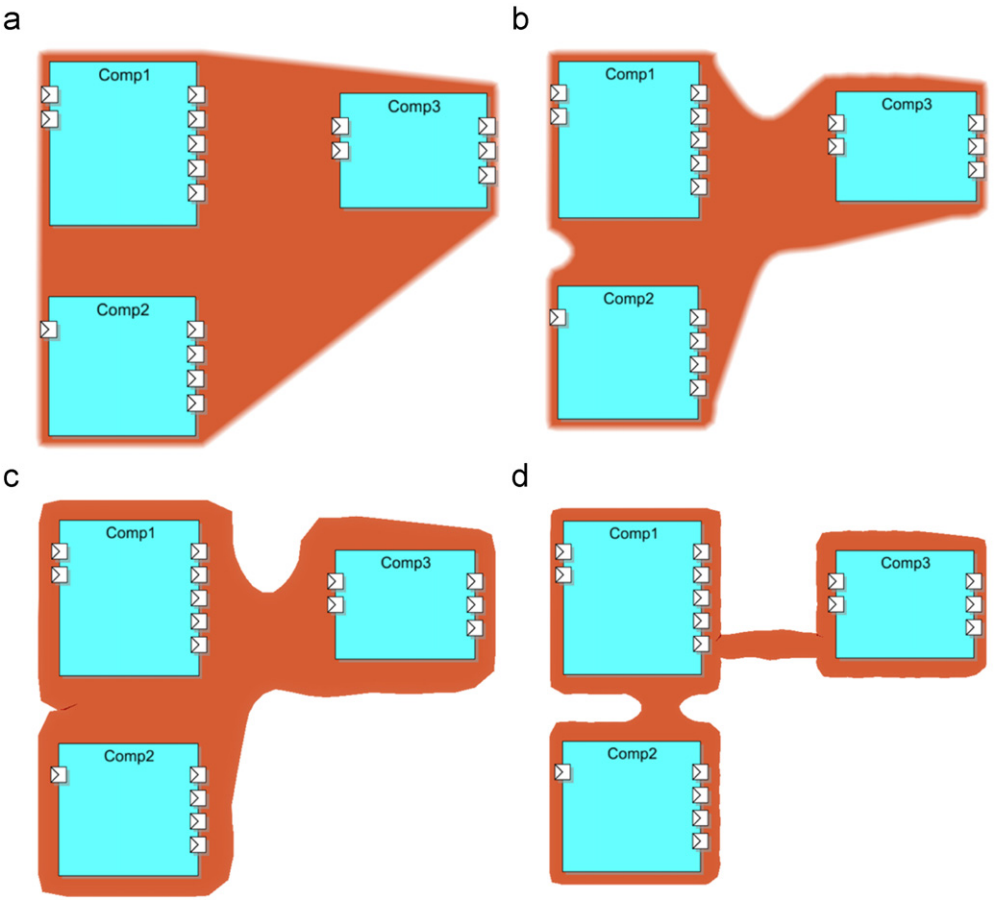

Fig. 3. Tightness and smoothness control. (a) 0 iterations, (b) 3 iterations, (c) 10 iterations, and (d) 20 iterations.

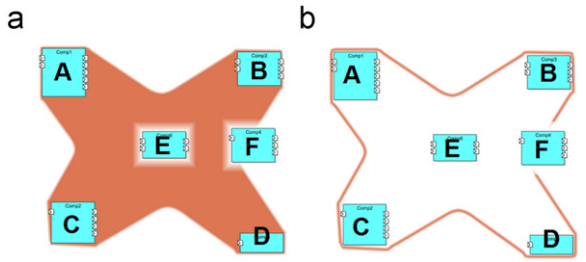

Fig. 5. Limitations of the eraser technique. When drawing the AOI as a contour, element *E* is incorrectly shown as being inside. (a) Filled drawing and (b) contour drawing.

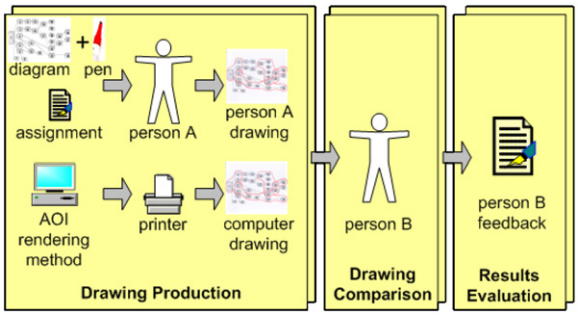

Fig. 6. Evaluation process.

completely inside the area, e.g. *E* in Fig. 5a. However, even in this case the eraser cue is not salient enough to easily see that *E* is outside the area. For elements partially overlapping the area which need to be excluded, e.g. *F* in Fig. 5a, the cue is even weaker. For contour-drawn areas, the eraser technique works very weakly (*F* in Fig. 5b) or not at all (*E* in Fig. 5b) as there is little or nothing to erase, this is a serious limitation since contour drawing is preferred to filled drawing in many situations, e.g. when one has only few available colors, when printing contours in black and white, when many contours overlap, or when blending-capable graphics hardware is not available. However, the most important problem of the eraser technique was that it turned out to be very unnatural for the most users who were shown it during our evaluation study (Section 5).

4.2. Unnatural flow-of-hand

Fig. 5 also shows a second problem of the original AOI-rendering technique. Close to the elements, the contours are too tight. In the middle, they are too loose. Also, the contour smoothness is not optimal. The contour looks too much like a sharp-angled polyline. The problem is only made worse by the eraser technique which introduces extra angles (e.g. element *F* in Fig. 5). This image is clearly different from user-drawn contours, which are much smoother (see, e.g. Fig. 22). The non-uniform tightness and angled look of the contours create a computer-made, unnatural look, quite different from the *flow-of-hand* typical to human drawings. This observation is in line with the previous research that has shown that (software) designers may prefer informally sketched designs and annotations to formal diagram elements [35,36].

5. Evaluation of the AOI-rendering method

At this point of our work, several questions were raised. We were aware of some limitations of the AOI drawing technique (see Section 4). Yet, how bad would real users find these in practice? Do other limitations exist which we were not aware of? And above all, what are the qualities of an AOI drawing that users would like the most, and how to simulate these in a computer rendering? Answering these questions is needed to estimate (and improve) the acceptance and usability of the AOI technique in practice.

To this end, we designed and executed an empirical evaluation. Thirty users of higher computer science education levels (master, post-master, Ph.D., and senior software designers) were selected. Some users were enrolled at the Eindhoven University in the Netherlands. Some others were part of an industrial European research project [37] where visualizing trust-related AOI on UML and component diagrams were a key task. All users had good knowledge of UML and had worked before for at least a few months (up to a few years) with class diagrams in daily software design activities. The evaluation flow is depicted in Fig. 6. It consists of three stages: drawing production, drawing comparison, and results evaluation. These stages are discussed in the next sections.

5.1. Drawing production

In the first phase, the participants were given a complex class diagram with 110 classes marked by numbers, printed in black-and-white on an A4 paper (Fig. 7) and seven AOIs, each given as a list of class numbers, printed on a separate paper. The list is as follows:

- Area 1: 4, 5, 13, 12, 17.
- Area 2: 51, 52, 55, 58, 59, 57, 68.
- Area 3: 14, 21, 22, 32, 40, 41, 60, 58, 59, 80.
- Area 4: 33, 34, 35, 36, 43, 61, 62, 66.
- Area 5: 66, 82, 81, 95, 96, 103, 105, 104.
- Area 6: 49, 50, 51, 67, 69, 111, 76, 78.
- Area 7: 86, 92, 93, 99, 94, 80, 96, 95, 103.

The participants were next asked to draw the areas as contours on this diagram, with a red marker pen we provided ourselves. The subjects were told that the goal of the drawing is to accurately and quickly convey, to another person, which class is in which area(s), and which area contains which classes. An example drawing, done on a different, much smaller, UML diagram containing 10 classes and one AOI, was also provided for basic illustration purposes. The complete experiment instructions were also provided on a separate A4 sheet. The subjects were also given a few paper sheets to practice on, before producing the final drawing. No verbal indications were

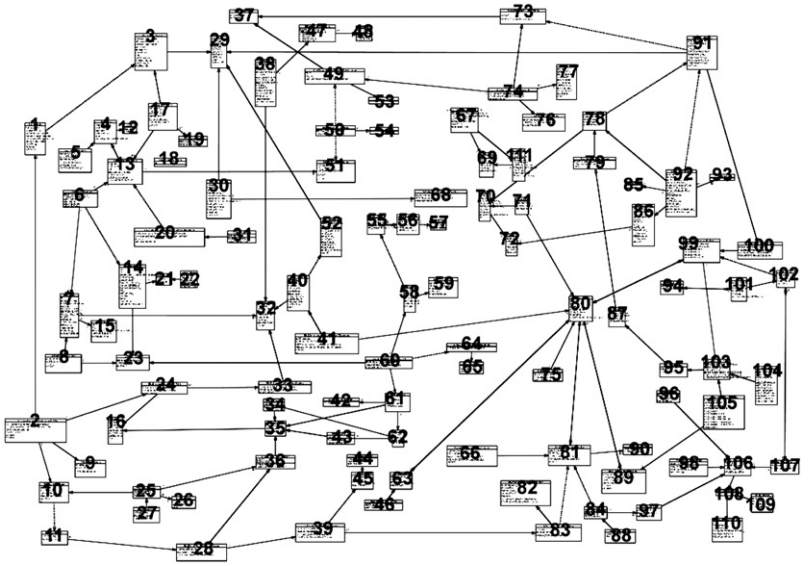

Fig. 7. Class diagram used in the evaluation.

given during the actual work, which lasted approximately 15 min. The subjects were not supervised. Also, they all worked independently, and had no knowledge of, or access to, the results of other participants.

Fig. 22a shows a scan of the drawing done by one of the participants.

Apart from the drawings made by the participants, and without their knowledge, we also produced a computer drawing on the same class diagram using the AOI-rendering method described in Section 3. We adjusted the algorithm and rendering parameters (e.g. line thickness and color) to look as similar as possible to the human drawings, and then printed the computer drawing on a similar sheet of paper. The result of the computer-drawn AOI is shown in Fig. 22b. Essentially, the only salient difference between the computer and human drawings is the shape of the contour.

5.2. Drawing comparison

In the second phase, we collected the results, and gave to each participant two drawings: a randomly picked drawing of another participant, as well as our unique computer-rendered drawing (Fig. 22b). Without telling which is which and without giving any hint that one of the drawing was computer-made, we asked the participants to complete a questionnaire (available in [38]). The questions included the following:

1. Rank the ease of understanding of the areas in each drawing on a scale of 1 (hardest) to 5 (easiest), accordingly to a Likert scale [39].
2. Which is the most complex area to understand?
3. Rank the perceived similarity between the two drawings on a scale of 1–3.
4. List, in plain text, what you liked least in the given drawings.
5. List, in plain text, what you liked most in the given drawings.

In the questionnaire, we mentioned that the main quality of an AOI drawing is given by its understandability, which is further related to its purpose. That is, the drawings should clearly show which area contains which classes, and which class is in which area(s). As mentioned in the previous section, the purpose of the area drawings was also discussed with the participants before the experiment, in order to be sure that they understood the use of such drawings.

5.3. Results evaluation

In the third phase, the questionnaire data were analyzed and aggregated. After collecting the questionnaires, we also had some short discussions (10–15 min) with the participants, in which we let them freely present their impressions and explain their results, and silently recorded their observations in writing. The results of this phase are summarized in the table in Fig. 8.

Several points become apparent now, as follows.

Most users found the machine-generated drawing (M) to be comparably understandable to the human-made one (H). Yet, the human drawings were almost always found to be better than the machine-generated ones, i.e. in 29 out of 31 cases (94%) (Fig. 8, column A). The perceived similarity between the machine and hand-drawn AOIs (Fig. 8, column D) showed a larger spread among users: 19 out of 31 (61%) marked a 2 (“not so different”), 10 users (32%) marked a 1 (“very different”), and the remaining two users (6%) marked a 3 (“very similar”). This can be explained by the relatively large variability of the different human drawings involved, and also in the fact that we refrained from providing similarity criteria to the users, to limit any potential biasing of the other assessments.

#	Better drawing (H or M)	Human quality (1-5)	Machine quality (1-5)	Perceived similarity (1-3)	Most complex area	Confusing overlaps	Exclusion unclear	Nonuniform tightness	Lacking flow-of-hand
1	H	5	3	2	-	+			+
2	H	4	3	2	3		+		
3	H	4	2	1	-			+	+
4	H	3	3	2	3	+	+		
5	H	4	3	1	2		+	+	+
6	M	3	4	1	3	+	+		
7	H	3	2	2	2	+	+		+
8	H	3	2	1	3		+	+	+
9	H	5	3	2	2	+			
10	H	4	3	2	3	+	+		
11	M	3	4	2	2	+	+		
12	H	4	3	2	3		+	+	
13	H	5	4	2	2		+		+
14	H	5	2	1	2	+	+	+	
15	H	5	2	2	3	+	+		
16	H	5	3	2	2	+			
17	H	5	2	1	2	+			+
18	H	4	3	2	3	+			+
19	H	5	2	1	2	+	+		+
20	H	5	4	2	-	+			+
21	H	4	2	3	5	+			
22	H	4	3	1	-	+	+		+
23	H	4	2	2	3		+	+	
24	H	4	4	3	3	+		+	+
25	H	4	2	2	2	+	+		
26	H	4	2	1	3	+	+		+
27	H	5	1	1	7	+	+		+
28	H	4	3	2	3	+	+		
29	H	3	2	2	7	+			+
30	H	4	3	2	3	+			+
31	H	5	4	2	2		+		+
	A	B	C	D	E	F	G	H	I

Fig. 8. Results of drawing comparison. Columns A–E indicate the following: which drawing was overall found to be better (human or machine), perceived quality of the human drawing, perceived quality of the machine drawing, perceived similarity of the two drawings, and visually most complex area. Columns F–I indicate often-perceived drawbacks of the machine drawings.

It is interesting to see that there is only a weak correlation between the perceived difference (column D) and the numerical difference between the perceived human and machine drawing qualities (columns B and C). Users that perceived their two drawings as being very different (value 1 in column D), e.g. rows 3, 5, 6, 8, 14, 17, 19, 22, 26, and 27 would score absolute human–machine quality differences of 1 (4 users, or 40%), 2 (2 users, or 20%), 3 (3 users, or 30%) and, respectively, 4 (1 user, or 10%). The two users that perceived their respective human and machine drawings to be very similar (score 3 in column D) ranked their human and machine drawing qualities to be 4 and 2, and 4 and 4, respectively. Finally, the three users who indicated the highest human and machine drawing quality scores (5 and 4, respectively) all indicated a perceived difference of 2 between their drawings. Overall, the emerging impression is that similar drawings are not essentially implying the same drawing quality (from the perspective of the indicated AOI goals), nor would a similar quality in two different drawings imply that they perceptually look the same.

The hardest-to-grasp (most complex) areas were quite consistent, i.e. Areas 2 and 3 (Fig. 8, column E). This also matches our opinion, and gives further an indication that the drawings done by different users are of comparable quality concerning understandability. Among different drawbacks of the machine-drawn areas found during the results analysis, two were most frequently named. The first drawback concerns the *eraser* technique (Section 4). The eraser, used to mark elements overlapping an AOI contour but not logically part of that AOI, is not working well, as we indeed suspected beforehand. We call this the *wrong exclusion* problem. For example, class 56 is not part of Area 2, as it is wrongly suggested by the computer drawing. This is clearly visible in Fig. 9, which shows a zoomed-in detail from the diagrams in Fig. 22. Fig. 9a, drawn using the AOI-rendering algorithm, does not show the AOI correctly. Fig. 9b, done by a human, is however

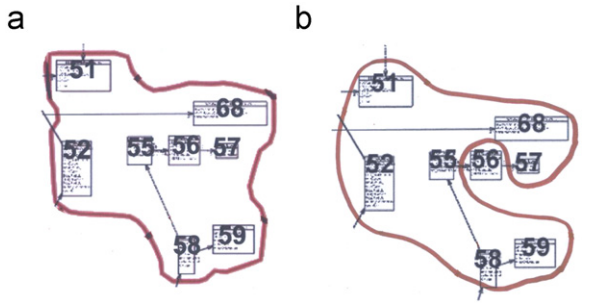

Fig. 9. Element 56 is not part of the area, as correctly shown in the human drawing (b). The eraser-based technique incorrectly shows 56 as inside the area.

correct. This problem was found by most subjects, as reflected in column G of the table.

The second drawback of the machine-generated areas concerns the contours' *tightness and smoothness*. These were perceived as being unpleasantly non-uniform (column F), and the *flow-of-hand*, i.e. similarity to the way humans draw, was lacking (column G). All users mentioned these aspects as hindering the drawings' understandability. As a third drawback, many subjects found the computer-drawn areas' *overlaps* confusing (column D). Contours which are near-tangent close to their intersection points were consistently named hard to understand in the light of the posed questions (Section 5.2). This was something we did not expect beforehand.

Clearly, there was room for improvement. The collected results point clearly in an overwhelming preference of the users for the human drawings, a fact which is also supported by the vast majority indicating higher or equal quality scores for the human drawings. It is natural to believe that a part of these differences are also reflected by the above-mentioned drawbacks of the machine drawings. In other words, removing some of these drawbacks has the potential of increasing the quality of the machine drawings. After analyzing the mentioned drawbacks, we designed several algorithmic improvements to the original AOI-rendering method to address them. These improvements are presented next.

6. Algorithm improvements

In Section 5.3 we identified three main problems of the computer-drawn AOIs: wrong exclusion, non-uniform tightness/smoothness, and confusing overlaps. We present next several algorithm improvements that address the first two problems.

6.1. Improved exclusion

An important problem, discussed in Section 4 and clearly visible from the empirical evaluation (Section 5), was the ineffective exclusion of elements overlapping with AOIs. In the previous work [9], the route of geometrically eliminating overlapping elements by modifying the contour is not followed, as it is deemed too complex to do for general diagrams. However, we have examined this direction and found a working solution. Our idea is to edit the contour, before deforming it, in order to exclude the wrongly overlapping elements. The process works as follows (see also the scheme in Fig. 10 and Listing 1).

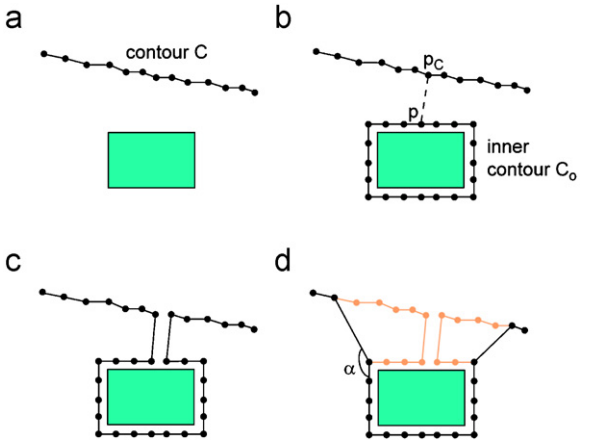

Fig. 10. Geometric-based exclusion steps. (a) Start situation, (b) find cut line, (c) connect contours, and (d) cut sharp corners.

Listing 1. Iterative exclusion algorithm

```
compute overlapping element set O = {o_i}
for (all o_i in O)
{
    create contour piece C_o around o_i
    p_o = point on C_o closest to contour C
    p_C = point on C closest to C_o
    ready = false

    while (!ready)
    {
      //Move inner point left
      p = p_o
      while (d(p, p_o) < d_max)
      {
        if (pp_C intersects no element e_j)
        {ready = true; break}
        move p to left along C_o
      }
      if (ready) break;

      //Move inner point right
      p = p_o
      while (d(p, p_o) < d_max)
      {
        if (pp_C intersects no element e_j)
        { ready = true; break }
        move p to right along C_o
      }
      if (ready) break;

      //Inner move failed, do outer move
      move p_C to left along C
    }

    connect C_o to C using line pp_C
    cut sharp corners
}
```

First the overlapping element set $O = \{o_i\}$ is computed by testing, for all elements, if any of the four element corners falls within the already computed AOI convex hull C (Section 3). This requires a simple point-in-convex-polygon test which is fast and robust. Next, each element o_i in O is excluded in turn, as follows. A finely sampled rectangular contour C_o is constructed around the bounding box of o_i (Fig. 10b). Next, a short *cut line* connecting C_o with the original contour C is computed such that it does not intersect any of the elements e_i in C. To do this, we use the following heuristic. We find first the closest two points $p_o \in O$ and $p \in C$. Next, we move both p_o and p_C along the inner and outer contours O and C, respectively, until the line does not intersect any element. We start by moving p_o around O to the left (counter-clockwise sweep) until a non-intersecting line is found or a too high distance d_{max} from the starting position, as computed along O, is reached. If no line can be drawn, we try now moving p_o to the right (clockwise sweep). If this fails too, then we move the other point p one step along the outer contour C, and repeat the inner contour sweep again. When a cut line was found (dotted line in Fig. 10b), we connect the inner and outer contours by constructing two sampled line segments, close and parallel to the cut line (Fig. 10c).

6.1.1. Limitations and workarounds

There are situations when the above heuristic (or any other algorithm, for that matter) cannot find a cut line that connects the element to be excluded with the AOI contour without hitting some other element contained in that area. This occurs, e.g. when the element to be excluded is completely surrounded by a ring of elements which are in the area (Fig. 12). This situation can be easily detected algorithmically by monitoring when point p_C has executed a full loop over the area's contour C (line 32 in the above listing). Although such configurations would not be typically found in many software architecture diagrams, we discuss below two methods to handle it.

The first solution groups all elements in O from an Area A which cannot be excluded using the cut technique in a new AOI A_{excl}. Since these elements are completely blocked from seeing A's contour, they must be fully contained in A. We now show the exclusion of these elements from A by drawing the contour of A_{excl} using the standard AOI algorithm, i.e. taking care to exclude elements which are in A but not in A_{excl}. Next, we draw the contour of A ignoring the exclusion. Fig. 12 (left) shows an example. The Area A logically contains the elements 1–5 but not the elements *Excl*1 and *Excl*2. The latter two cannot be handled by the cut line technique, so they constitute A_{excl}. Hence, we draw the contour of A ignoring *Excl*1 and *Excl*2 and separately the contour of A_{excl}, this time taking care to avoid element 5, which is not in A_{excl}. The A_{excl} contour is nested in the A contour, since the elements in A_{excl} are completely within A. This technique handles well a large range of configurations. However, it would fail when the inner contour drawing (A_{excl}, which involves the cut line method, fails from precisely the same reason the initial exclusion of its elements from A failed. This happens in configurations involving *several* concentric rings of elements which alternately belong to A and A_{excl}. Although we could handle such situations by the addition of more contour pieces, this would create AOIs with several disconnected components which are increasingly hard to follow visually.

From discussions with actual software engineers who use UML and tested our tool on repeated occasions, we observed a net preference for rendering AOIs as simply connected contours (shapes without holes) rather than multiple (inner and outer) contours. The main explanation given was that simply connected shapes are easier to follow visually, especially in complex diagrams with several areas whose contours overlap. In such cases, one needs to visually follow an area's contour to discern that area, so an area with multiple disconnected boundaries may be wrongly perceived as several separate areas. Secondly, there are cases when one avails a few (or no) colors to draw the areas, e.g. in print-outs. In such cases, it is hardly possible to group disconnected contours as inner and outer boundaries of the same area.

A second solution for the cases when the cut line algorithm fails due to blocked contour visibility is to use as cut line the shortest segment linking the element(s) to exclude with the AOI's contour, i.e. the segment (p_o, p_C) computed as in lines 5 and 6 in Listing 1. Although this cut will intersect (at least) one of the elements which are logically inside the area, it has the desirable property of being the shortest possible one, thus the visually least disturbing, and also it generates a simply connected contour.

Fig. 12 (middle) shows such a situation. The AOI contains elements 1–4 but must exclude element 5, which cannot be connected via a straight path with the area's contour. Since no non-intersecting cut line can be found, the shortest cut line is used, which will intersect element 3. To further emphasize this cut, we skip the sharp corner cutting and smoothing (described in the next section) for this cut. The cut will, hence, stay thin and have a visually distinct appearance from the regular cuts (compare Fig. 12, left, with Fig. 11f). For comparison purposes, Fig. 12 (right) also shows the original eraser technique described in [9], which only works on filled areas. Overall, the preferences informally observed ranked the shortest-cut solution as the most accepted (Fig. 12, middle), followed by the multiple contours (Fig. 12, left) and finally the eraser technique (Fig. 12, right).

6.2. Natural flow-of-hand

Our experiment subjects found contours having sharp corners to be unnatural. Moreover, the contour editing presented in Section 6.1 can create contours having unnecessary sharp corners. We reduce these in a separate pass. For each point p_i along the contour, we compute the angle $\alpha = p_{i-1}\widehat{p_i p_{i+1}}$ made by that point with its two neighbors. If the angle drops below a minimal value α_{min}, and the line $p_{i-1}p_{i+1}$ does not intersect any diagram element e_i, then we remove p_i from the contour by connecting p_{i-1} and p_{i+1}. Good values for α_{min} are in the range [40,70] degrees. We repeat this procedure iteratively until no removal is possible. The final result is

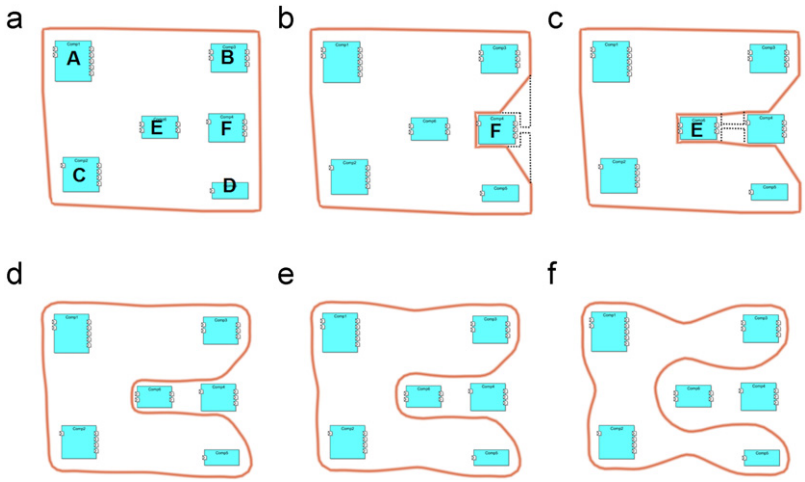

Fig. 11. Geometric-based exclusion. The convex hull is edited to iteratively exclude all overlapping elements (E, then F) and also cut sharp corners. After that, smoothing and shrinking take place. (a) Convex hull, (b) exclude F, (c) exclude E, (d) smooth (1 step), (e) smooth (2 steps), and (f) smooth (5 steps).

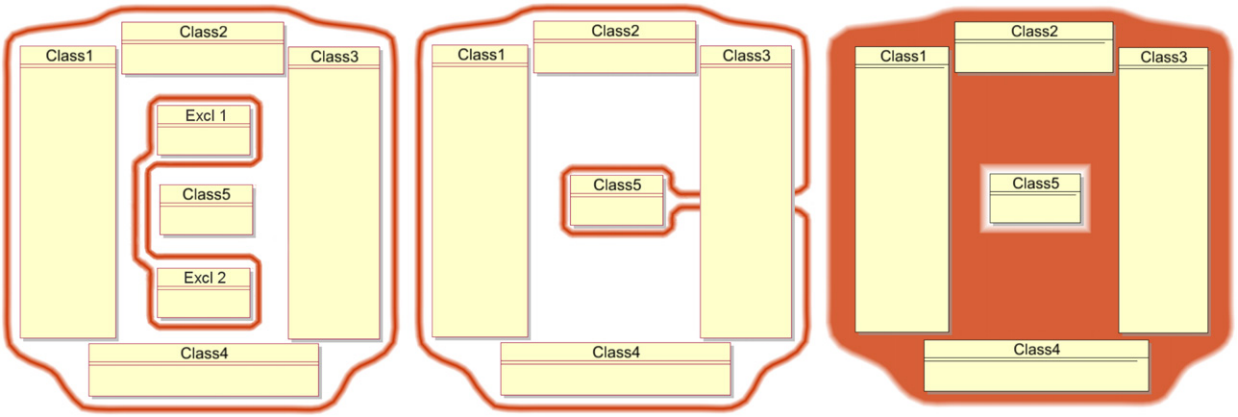

Fig. 12. Exclusion algorithm for cases when no straight cut from the excluded element (5) to the area's contour is possible. Shortest-cut solution (left) and eraser solution (right).

shown in Fig. 10d. When excluding several overlapping elements o_i, sharp corners are removed after excluding each element o_i, and not at the end. This gives better quality, as unnecessary sharp corners are eliminated as soon as possible. This also accelerates the further smoothing steps, since the contour gets simpler (less points). Finally, the shrinking is more stable if sharp corners are eliminated, a well-known fact from the geometric level set theory [40].

Fig. 11 shows the same diagram as in Fig. 5. We see how the elements F and E are iteratively removed (Figs. 11 b and c). The red line shows the contour after exclusion and sharp corner removal. The dotted black line shows the contour after exclusion but before sharp corner removal. Figs. 11d–f show the result after doing a few smoothing steps. Clearly, these results are better than the original one in Fig. 5b. The exclusion algorithm now very clearly shows what is inside, and what outside, an area. The unnatural eraser effect is now gone, and the contour resembles much more to what a human would draw. The sharp corner cutting procedure has the additional positive effect of smoothing the contour, yielding a more natural "flow-of-hand"-like drawing.

A final improvement of the original method targets the problem shown in Fig. 5, i.e. the fact that the drawn contours are too tight close to the elements and too loose in the middle. A more uniform tightness along the entire contour is preferred. We achieve this by the following preprocessing step. Right after the convex hull is sampled, and before the exclusion begins, we offset every point p_i on the contour C outwards with a small distance equal to the shrinking step ε_n (Section 3). Fig. 13a shows this process on a zoom-in at one of the corners of the diagram in Fig. 5. The offset makes the initial contour looser, which gives it further space to deform and nicely curve itself around the elements (Fig. 13b).

Fig. 14b shows the result of our improved method on the same diagram detail as in Fig. 9. We see that the

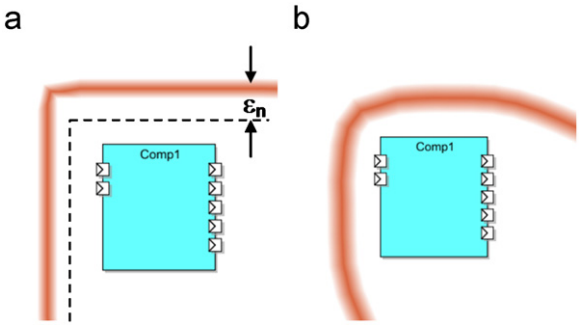

Fig. 13. Contour smoothing close to elements. (a) Contour offsetting and (b) smoothed contour.

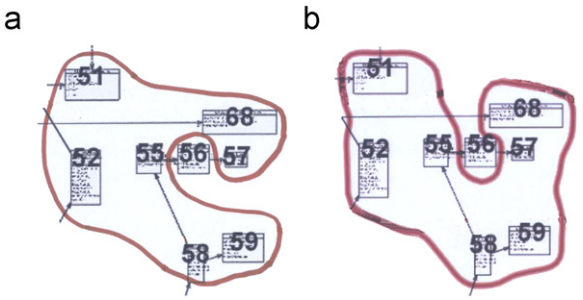

Fig. 14. Comparison of human drawing (a) with the improved rendering method (b). Details from Figs. 22b and c.

improved method (Fig. 14b) is more similar to hand-drawing (Fig. 14a) than the original method (Fig. 9a). Given the contour's initial construction from a fixed convex hull [9] and the smoothing constraints described above, the AOI shape has only a limited freedom to deform. Following the terminology of [33], we can classify our AOI annotations as having a medium-to-high level of formality.

7. Quantitative analysis

As the results of our user study showed (Section 5), the human-made drawings were perceived by users to be almost always better understandable than our computer-generated ones. We presented in Section 6 several algorithm improvements by which we hoped to address the shortcomings of our computer-rendering method.

However, how to measure how well we improved as compared to the original algorithm? Repeating the user study (Section 5) with the *same* audience could be biased, since the users by now knew our aims, diagram datasets, and already had some experience. Conducting the same study on a different group of subjects and/or different datasets could be done, but how to quantitatively compare subjective qualitative opinions of two different groups and/or datasets? Additionally, a user experiment does not give a precise, quantitative answer for how much closer or further our new algorithm improves the rendering.

If we were interested to test the suitability of the AOI renderings for a very specific comprehension task, e.g. the amount of time it takes to visually locate a given diagram element in a given area, we could indeed perform two user experiments to measure the time difference when using the improved, versus the original, rendering methods. However, there are several drawbacks to such an approach. First, as indicated by the user comments collected in our study, human-drawn AOIs have several quality attributes which help comprehension (e.g. the natural flow-of-hand). These are hard to quantify by means of, e.g. timing a single (or a few) narrow tasks. We do not yet know which tasks would be the most representative here. Our main assumption is that drawing AOIs *as humans do it* is good for comprehension, in line with the previous authors [31–33]. Hence, we would like to test that our improved algorithm produces drawings closer to human drawings than the original algorithm.

We designed a way which provides a quantitative answer to the above point. The quantitative analysis pipeline (Fig. 15) is described next.

Firstly, we extracted the area contours from all drawings, i.e. human- and computer-generated with both the original and improved algorithm. For this, we used a simple filter-by-color thresholding technique, which was reliable as the contours and diagrams were drawn with two predefined distinct colors, i.e. red, respectively, black. An example of the extracted contour is shown in Fig. 16, which is indeed a clean, noise-free, contour representation.

Next, we would like to measure the difference between any two contours, i.e. human- and/or computer-drawn. We do this as follows. Consider two contours C_i and C_j like the ones in Fig. 16. For a contour C, we denote by D the distance transform, or *distance map*, of C. The distance map is defined as

$$D(p) = \min_{q \in C} |p - q|, \quad \forall p \in \mathbb{R}^2 \tag{3}$$

for any point p in the 2D plane. Essentially, $D(p)$ gives the distance from any point p to the closest point q on the contour C. We know that the distance map of an object C is the solution to the so-called Eikonal equation:

$$|\nabla D| = 1 \tag{4}$$

with the boundary condition $D = 0$ on all points of C. We solve Eq. (4) using the fast marching method as described by Sethian in [40], on the same pixel grid as the one on which the scanned contour C is stored, and obtain the distance map D at a pixel-level spatial resolution. A careful implementation of the fast marching method ensures the distance map D is computed on an image of 1024×768 pixels in under 1 s on a 1.8 GHz Windows PC [41].[2] Fig. 17 shows the distance map D (using a blue-to-red colormap) of the contour C (shown in white).

Now, given a contour C_i and its distance map D_i, computed as above, we define the distance d_{ij} of C_i to

[2] The implementation of the fast marching method used is available at www.cs.rug.nl/~alext/AFMM.

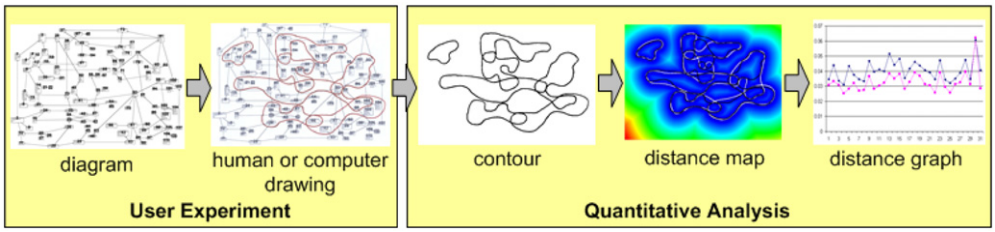

Fig. 15. Quantitative analysis process.

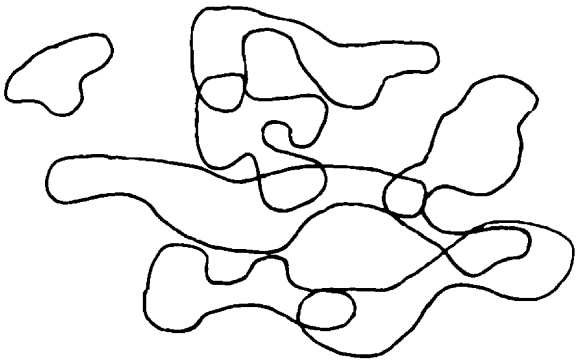

Fig. 16. Extracted contour.

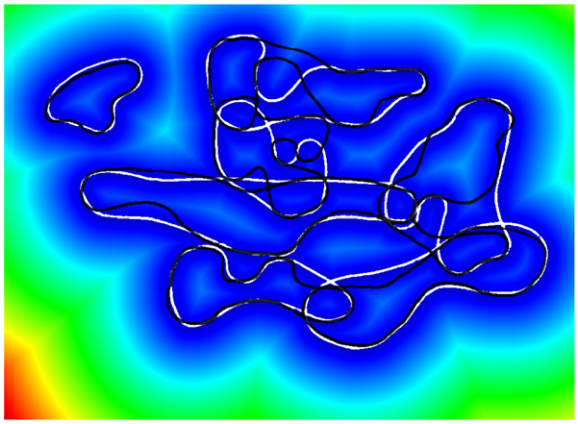

Fig. 17. Distance map D (blue-to-red colormap) of contour C (white), used to compare C with a second contour C' (black) (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article).

another contour C_j as

$$d_{ij} = \frac{1}{2}\left(\frac{\sum_{p\in C_j} D_i\ (p)}{|C_j| D_{i_{max}}} + \frac{\sum_{p\in C_i} D_j\ (p)}{|C_i| D_{j_{max}}}\right) \tag{5}$$

In the above, D_j denotes the distance map of contour C_j, while $D_{i_{max}}$ and $D_{j_{max}}$ are the maximum values of D_i and D_j, respectively, over the considered images. $|C_i|$ and $|C_j|$ denote the contour lengths in pixels. The above definition of d_{ij} ensures d is a symmetric function $d_{ij} = d_{ji}$, and also is normalized between 0 and 1. Intuitively, Eq. (5) states that the distance between the two contours C and C' is proportional with the area between the two contours, which is a perceptually good measure [42]. Alignment and image registration problems were, in our situation, not an issue, since all drawings were done on the precisely the same class diagrams, rendered on identical canvases, scanned at the same resolutions. However, the distance metric given by Eq. (5) is robust to small shifts and rotations. Let us stress that the proposed distance function is just one of the many ways in which we can compare contour drawings. Other more sophisticated measures, e.g. perceptual-based metrics [43], template-based matching [44], or contour matching using the earth mover's distance [45], can be used as well. We prefer to use our simpler metric since its behavior is easier to interpret and its implementation is quite straightforward. Also, using more complex distance metrics typically involves having a clearer idea of which features (e.g. angles, protrusions, concavities, and flat regions) are perceptually more important for the match, and how to quantify this importance, which is the information that we do not have at the present moment.

We can now build a matrix d_{ij} containing all distances between any two contours of the 31 hand-drawn ones plus the two computer-drawn ones (with the original, respectively, improved, methods). However, as the user evaluation results showed (Fig. 8), not all hand drawings are found to be of the same quality. We are in particular interested to see how our computer-drawn contours compare to the good human drawings and, also, if the proposed improvements did, indeed, bring us closer to these drawings.

For this, we first split the 31 human drawings into three groups: good, average, poor, based the "human quality" scores of 5, 4 and 3, respectively (see Fig. 8). Fig. 18 shows the distances of the six good drawings (H1 ... H6) to themselves and to the computer-generated drawings with the original (OLD) and improved (NEW) methods. We see that all drawings in this table are quite similar to each other. We also see that the NEW drawing is consistently closer to the human drawings than the OLD drawing.

Fig. 19 graphs the distances between all 31 human drawings and the two (initial and improved) computer drawings. The human drawings are grouped according to their perceived quality, as described above. Several observations can be made here. First, there is some variation in the distances within the same quality class. This is expected, since each quality value was assigned subjectively by just one person.

	H1	H2	H3	H4	H5	H6	OLD	NEW
H1	0	0.0235	0.0249	0.0212	0.0261	0.0243	0.0345	0.0307
H2	0.0235	0	0.0368	0.0340	0.0229	0.0320	0.0440	0.0334
H3	0.0249	0.0368	0	0.0198	0.0333	0.0230	0.0335	0.0308
H4	0.0212	0.0340	0.0198	0	0.0267	0.0268	0.0311	0.0253
H5	0.0261	0.0229	0.0333	0.0267	0	0.0319	0.0433	0.0283
H6	0.0243	0.0320	0.0230	0.0268	0.0319	0	0.0377	0.0321
OLD	0.0345	0.0440	0.0335	0.0311	0.0433	0.0377	0	0.0226
NEW	0.0307	0.0334	0.0308	0.0253	0.0283	0.0321	0.0226	0

Fig. 18. Distance table of the best human drawings and AOI rendering.

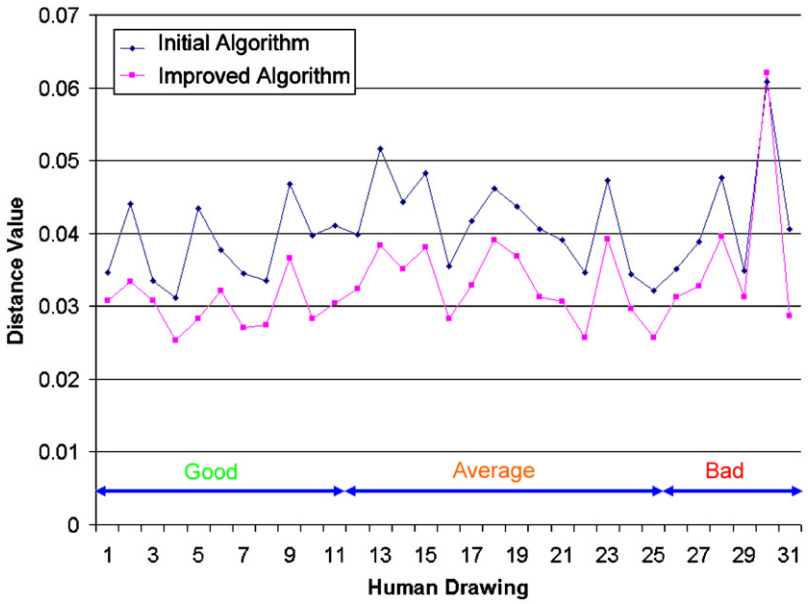

Fig. 19. Comparison of AOI-rendering algorithms. The improved rendering method (bottom graph) yields results closer to the human-drawn AOIs than the original rendering method (top graph).

However, the distance values, averaged per quality class (Fig. 20), show that the computer-generated drawings are *closer* to good than to bad drawings, both for the original and improved methods. Also, we see that the improved method brings the computer-generated drawings closer to the human drawings in all quality classes as compared to the original method. The improvement is of roughly 20 percent for the "good" and "average" classes and 12 percent for the "bad" class. This is also visible from the graphs in Fig. 19. More importantly, the improved method brings the computer drawings closer to the good human drawings than to the bad ones.

Finally, we notice an outlier: the human-drawing 30 in Fig. 19. Looking at it (Fig. 21), we can indeed see it has a very different style from a typical good human drawing (e.g. Fig. 22a). Also, the user who made drawing 30 forgot to draw one area (Area 6 - see Section 5), which explains the high spike in the distance metric. This shows that our distance metric is quite discriminative in the presence of erroneous drawings.

8. Discussion

8.1. Comparison with original method

To discuss our findings, let us consider three drawings of AOI on the diagram used in our evaluation (Fig. 22): The top image is an actual scan of one of the best, most understandable, human drawings. The middle image shows the result of the original AOI drawing method [9], as it was shown to the evaluation subjects. In this drawing, we recognize all problems named so far: some elements (*A*, *B*, and *C*) are incorrectly included in surrounding areas, whereas they should be outside, as shown in the top drawing; and the contours are tight and

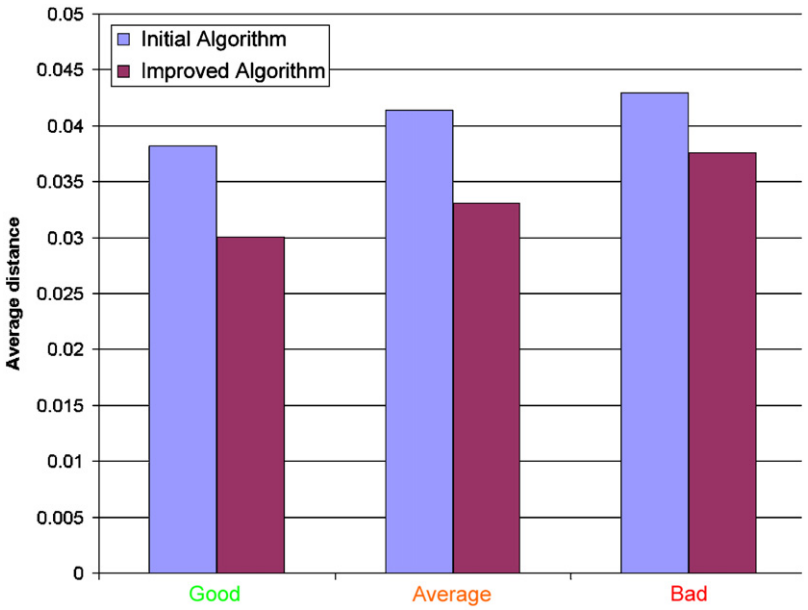

Fig. 20. Comparison of average distances between three groups of human drawings (good, average, bad) and two computer-generated drawings (initial and improved).

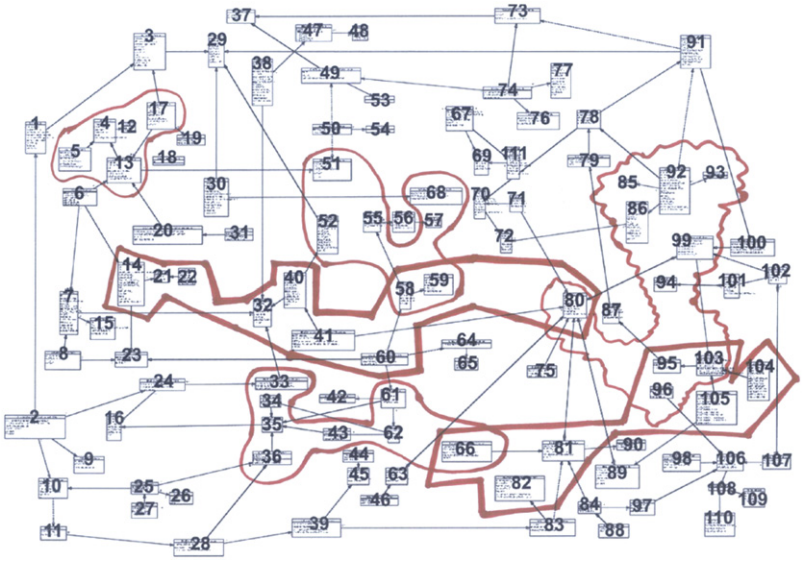

Fig. 21. Atypical human drawing.

sharp close to the elements but loose and smooth in the middle. The bottom drawing shows our improved algorithm. The elements *A*, *B*, and *C* are now correctly excluded. The contours have a more uniform smoothness and are not so tightly close to the elements. We found this drawing to be of a higher quality as compared to the one produced by the original algorithm. Probably the most appealing fact is that the areas drawn here simply *look* natural and quite similar to the human-drawn ones.

Fig. 23 shows another example featuring 12 areas on a different class diagram with 110 classes, this time drawn as filled contours. The right zoom-ins show the areas framed in white on the left images. We see now the two problems of the original method again: Class *A* is incorrectly marked as contained in Area 1. This is because the eraser (Section 4.1) is overwritten by the color of Area 2, in which *A* is indeed included. Class *B* is only in Area 2, so the eraser is visible as a faint white border. However, when the diagram is zoomed out, this eraser becomes almost invisible. The improved method (Section 6) remediates both problems. Now the inclusion of *A* in Area 2 and the fact *B* is outside both Areas 1 and 2 is clear.

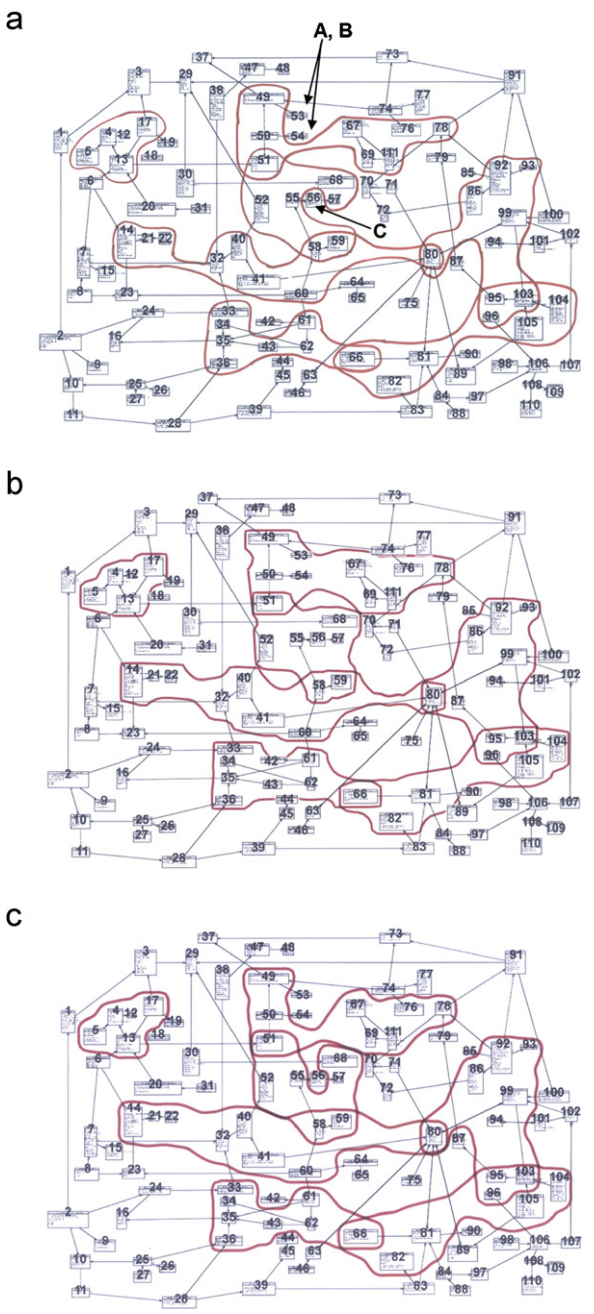

Fig. 22. Comparison of AOI drawings. (a) Human drawing, (b) original method, and (c) improved method.

The AOI-rendering technique can be used on different types of diagrams besides class diagrams. Fig. 24 shows two AOI rendered on a message sequence chart. Here, the new exclusion technique is crucial, due to the typical layout of such diagrams. In [46], AOIs are used to show performance-related parameters on a dataflow diagram of a component-based system. Generally speaking, the AOI-rendering technique can be used to emphasize logical subsets on many types of graph-like diagrams.

The improved method does not introduce new parameters that the user has to explicitly tune. The main user parameters, i.e. AOI color, drawing mode (filled or contour), AOI transparency, and contour tightness, are still the same as in the original method. In particular, the relatively complex geometric exclusion algorithm is fully automatic. The speed of the improved method is slightly (about 10–15%) worse as compared with the original method. This is due mainly to the exclusion process which has to find an optimal cut line (Section 6) and also cut corners, both being iterative processes. However, we should stress that we have not highly optimized this code. Standard geometric optimizations such as spatial search techniques [47] can easily eliminate this performance decrease. Even with the performance drop, the AOI rendering still occurs in subsecond time.

8.2. Incorporating edges in AOI

In some cases, it may be desirable to constrain not only elements of a diagram to be contained in a given AOI, but also edges denoting relationships between such elements. For example, if two elements e_1 and e_2 are contained in an Area A, then one may desire to constrain A to also include all edges (e_1, e_2) between them. A typical use-case hereof is emphasizing structural patterns such as design patterns. This can be achieved by adding sample points on the edges whose both end-elements are contained in a given AOI to the set of sample points generated on the elements' bounding boxes. The AOI construction algorithm will treat all these points uniformly, i.e. deforming the contour and constructing the cut lines without getting too close to them. When using straight-line segments to represent edges, these are contained by default in the initial convex hull that encloses all the area elements, so the only difference is the increased number of sites (points) taken into account during the area deformation and exclusion step. Note that precisely the same technique can be used for handling elements with non-rectangular shapes.

This technique works very well for diagrams having areas with little overlap. Consider, for example, a class hierarchy drawn on several layers as a tree or directed acyclic graph. The elements' convex hull will automatically include all edges in this hierarchy. The deformation (shrinking) step will move the contour inwards, as usual. If sample points on these edges are explicitly added to the AOI, the contour shrinking will stop when getting in their proximity, the rest of the algorithm staying unchanged.

However, when several areas considerably overlap and they also contain many edges, the blocking configuration described in Section 6.1.1 may occur more frequently. Such situations can generate overly complex curved contours, which, again, are hard to follow visually. An alternative solution to handle the logical edge inclusion in AOIs is to refrain from geometrically including them in the areas, but mark them with the color(s) of the area(s) they are part of. Though not ideal, this solution is robust and easy-to-implement.

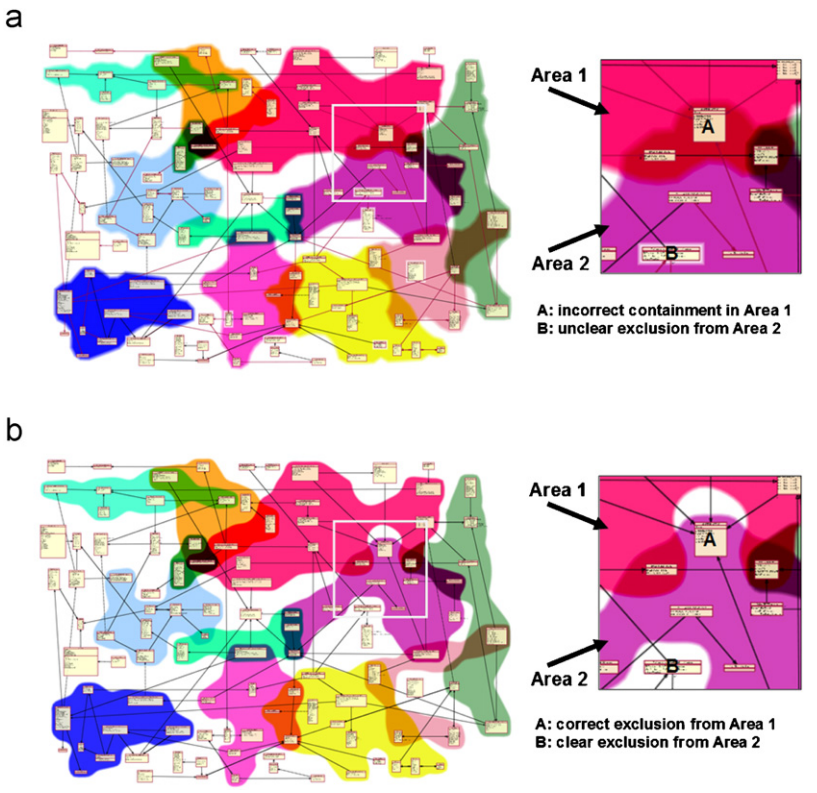

Fig. 23. Comparison of filled AOI drawing. The improved method removes several problems of the original method. (a) Original method and (b) improved method.

8.3. Limitations

Although the improved AOI drawing method yields better appreciated drawings, which are measurably closer to human drawings than the original method, it still has some limitations. First, our users have found near-tangently intersecting contours to be hard to understand (Section 5.3). We have not addressed this problem yet. For this, we should consider a global contour rendering rather than the per-contour rendering we use now.

Besides a possible optimization of the contour crossing angles, a global contour construction could also optimize the actual positions of contour points, in order to pull apart AOI fragments which are too close. Although this direction should be explored in future work, it introduces some important problems. In typical usage scenarios, one needs to switch areas on and off interactively during analysis. If the shape and position of an AOI are influenced by other AOIs, then toggling on or off an Area A_i may abruptly change the look of another Area A_j in the same drawing, which is highly disruptive. Precomputing all AOI contours beforehand is not an option, since a typical diagram can easily have tens of areas showing different concerns (performance, resource usage, reliability, coding and documentation aspects, and so on), and we do not know in advance which areas one may want to visualize at a time. Finally, a global optimization may incur performance problems on large diagrams with many areas.

8.4. Human–machine drawing comparison

The distance metric proposed to compare contours is well known in shape analysis and computer vision applications (see, e.g. [42]). Its main advantage is good robustness to small-scale geometric noise, and rotation and scale invariance. However, it does not take into account specific quality attributes for the tasks related to AOI. For example, we can argue that a small geometric difference between two contours is perceptually more important if located at some point where several contours overlap or intersect, or in an area covered by a complex diagram, than at the periphery of the drawing. Integrating perceptually driven distance metrics [43,44] in our evaluation should lead to further insights.

Finally, we are aware that we have not conducted a formal user experiment, i.e. a quantitative measurement of the (in)validation of a hypothesis. Our evaluation's main goal was to harvest information about the differences perceived between computer- and human-drawn AOIs, and to adapt our computer drawings accordingly. Assuming that our hypothesis holds that human-drawn AOIs are easy to understand, we argue that our improved rendering algorithm produces better drawings, since these

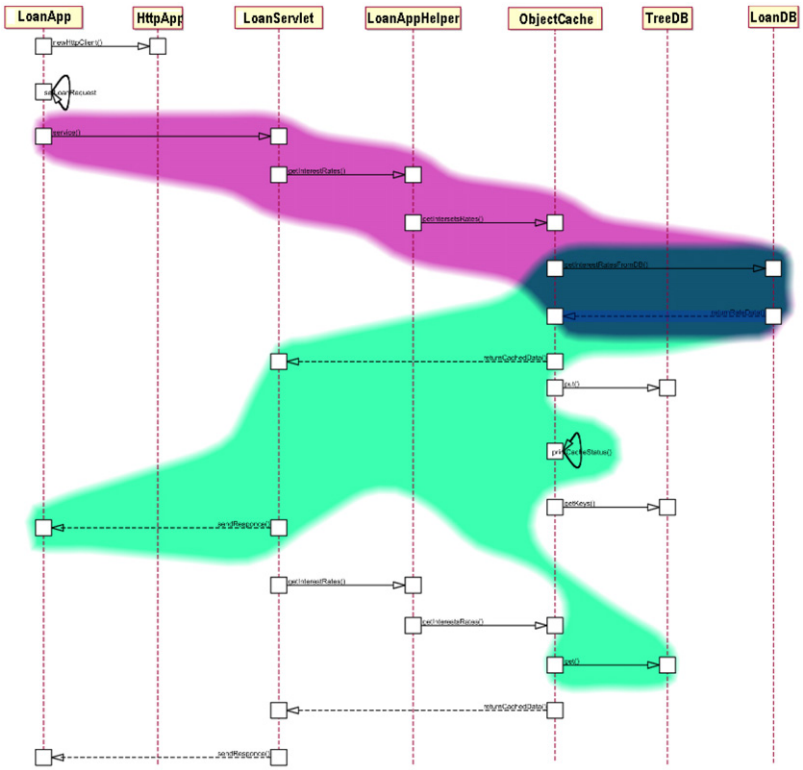

Fig. 24. Example of drawing AOIs on message sequence diagrams.

are measurably closer to human drawings than the original computer drawings.

The main advantages of the machine drawings, as compared to the human drawings are as follows:

- Rapid and automatic handling of complex areas on large diagrams. A human will typically take several minutes to draw a set of areas, while the algorithm presented here needs a few seconds.
- Correct drawing. As shown by the experiment, humans can draw incorrect areas, especially for complex configurations (see, e.g. Fig. 9).
- Easy parameterization of the drawing process (contour smoothness, color, thickness, filled or not).

The main elements which still need to be incorporated in the automatic algorithm, as outlined by the user study, are as follows:

- *Contour crossings*: Although the improved algorithm presented here significantly reduces sharp angles and produces overall smooth contours, it can still produce contour crossings having small angles. These crossings can be detected, and an additional force can be added to the contour points in the crossing's vicinity to maximize these angles in the shrinking process.
- *Contour separation*: Since contours are drawn independently, they can have (near) overlapping fragments, which are hard to separate visually. This could be addressed by a separate relaxation pass: After all contours are constructed independently, repulsion forces are added to the contour points, and several deformation steps are performed. Alternatively, the contours can be drawn iteratively, and the repulsion forces can be added to each newly drawn contour as it is shrunk.

9. Conclusions

We have presented our attempt to computer-rendered areas of interest (AOIs) on UML diagrams so they resemble easy-to-understand human drawings. First, we compared an existing AOI-rendering method [9] with human-drawn results by means of an empirical evaluation, and found some aesthetic and correctness limitations of the existing method, as well as what users consider to be a good drawing. The method in [9] scored averagely in aesthetics and overall quality. Next, we designed several improvements to the original rendering method. The geometric exclusion technique creates contours which correctly and visibly exclude all elements. The corner-cutting technique, which eliminates sharp corners, and the inflation technique, which preprocesses the contour before shrinking, combine to create a smooth, evenly fitting, natural flow-of-hand drawing style. Our third contribution is a quantitative way to measure how close are the computer-generated renderings to the human drawings. We used this measure to show that our proposed improvements

effectively brought the computer-made drawings closer to typical (good) human drawings.

To our knowledge, our work is one of the first attempts to measure empirically and quantitatively the quality of UML diagram annotations, such as AOI, and to use these measurements to improve computer-generated annotations. We found that user studies, although highly time-consuming and laborious, are excellent and indispensable instruments to get deeper understanding of what makes a visualization good and accepted. In our case, there was very strong consensus among very diverse users, who did not communicate with each other, about what is nice and less nice in the AOI drawings. Designing the proposed rendering improvements followed naturally, once we understood what the users like to see in the drawings. Hence, we believe that we succeeded to address here requirements 3 (correctness) and 5 (measured similarity with human-drawn areas) from the desirable set of five requirements on AOI drawing (Section 2). Requirements 1 and 2 were already addressed by the original rendering method.

We next consider to design a computer-rendering method that mimics even closer (good) human drawings. A way to do this is to perform a machine-learning process that optimizes the proposed distance metric by varying the computer method's rendering parameters. Another direction is addressing requirement 4 (limited cluttering) on complex diagrams, by a combination of geometric, shading, and texture techniques.

Acknowledgments

This research was part of the ITEA project Trust4All, which develops methods to describe, evaluate, and assess trust in component-based middleware frameworks. This project was partly conducted while the authors were with the Eindhoven University of Technology, the Netherlands.

References

[1] O.M. Group, The unified modeling language, 2008 ⟨www.uml.omg⟩.
[2] IBM, Rational Rose modeling tool, 2007 ⟨www.306.ibm.com/software/rational⟩.
[3] Borland Inc., Together modeling tool, 2007 ⟨www.borland.com/together⟩.
[4] Telelogic, Telelogic Tau modeling tool, 2008 ⟨www.telelogic.com/products/tau⟩.
[5] ArgoUML. ArgoUML modeling tool, 2008 ⟨argouml.tigris.org⟩.
[6] N. Fenton, S. Pfleeger, Software Metrics: A Rigorous and Practical Approach, Chapman & Hall, London, 1998.
[7] N. Gill, P. Grover, Component-based measurement: a few useful guidelines, ACM SIGSOFT Software Engineering Notes, 28 (2003).
[8] J. Wust, SDMetrics: the software design metrics tool for UML, 2005 ⟨www.sdmetrics.com⟩.
[9] H. Byelas, A. Telea, Visualization of areas of interest on software architecture diagrams, in: Proceedings of the ACM SoftVis, 2006, pp. 105–114.
[10] A. Marcus, L. Fend, J.I. Maletic. 3d representations for software visualization, in: Proceedings of the ACM SoftVis, 2003, pp. 27–36.
[11] M. Termeer, C. Lange, A. Telea, M. Chaudron, Visual exploration of combined architectural and metric information, in: Proceedings of the VISSOFT, IEEE Press, New York, 2005, pp. 21–26.
[12] Q. Chen, J. Grundy, J. Hosking, An e-whiteboard application to support early design-stage sketching of UML diagrams, in: Proceedings of the VL/HCC, IEEE Press, New York, 2003, pp. 219–226.
[13] S. Tilley, K. Wong, M. Storey, H. Müller, Programmable reverse engineering, International Journal of Software Engineering and Knowledge Engineering 4 (4) (1994) 501–520.
[14] Prefuse, The prefuse information visualization system, 2007 ⟨http://prefuse.org⟩.
[15] E. Gansner, S. North, An open graph visualization system and its applications to software engineering, Software—Practice and Experience 30 (11) (2000) 1203–1233.
[16] J. Rilling, S.P. Mudur, On the use of metaballs to visually map code structures and analysis results onto 3D space, in: Proceedings of the WCRE, IEEE Press, New York, 2002, pp. 299–306.
[17] T. Sprenger, R. Brunella, M. Gross, H-blob: a hierarchical clustering method using implicit surfaces, in: Proceedings of the Visualization, IEEE Press, New York, 2000, pp. 61–68.
[18] M. Balzer, O. Deussen, Exploring relations within software systems using treemap enhanced hierarchical graphs, in: Proceedings of the VISSOFT, IEEE Press, New York, 2005, pp. 89–94.
[19] H.C. Purchase, D.A. Carrington, J.-A. Allder, Graph layout aesthetics in UML diagrams: user preferences, Journal of Graph Algorithms and Application 6 (3) (2002) 255–279.
[20] H.C. Purchase, D.A. Carrington, J.-A. Allder, Empirical evaluation of aesthetics-based graph layout, Empirical Software Engineering 7 (3) (2002) 233–255.
[21] H.C. Purchase, L. Colpoys, D.A. Carrington, UML collaboration diagram syntax: an empirical study of comprehension, in: Proceedings of the VISSOFT, 2002, pp. 13–22.
[22] H.C. Purchase, R. Welland, M. McGill, L. Colpoys, Comprehension of diagram syntax: an empirical study of entity relationship diagram notations, International Journal of Human–Computer Interaction 61 (2) (2004) 187–203.
[23] J. Krogstie, Evaluating UML using a generic quality framework, in: UML and the Unified Process, Idea Group Inc., 2003, pp. 1–22.
[24] A. Bobrowska, A framework for empirical evaluation of model comprehensibility, in: Proceedings of the SOFSEM, Springer, Berlin, 2005, pp. 72–81.
[25] J. Aranda, N. Ernst, J. Horkoff, S. Easterbrook, A framework for empirical evaluation of model comprehensibility, in: Proceedings of the International Workshop on Modeling in Software Engineering (MISE), 2007, pp. 7–15.
[26] O.I. Lindland, G. Sindre, A. Solvberg, Understanding quality in conceptual modeling, IEEE Software 11 (1994) 42–49.
[27] M. Page-Jones, What every programmer should know about object-oriented design, Dorset House Publishing, 1995.
[28] J. Rumbaugh, Notation notes: principles for choosing notation, Journal Object-Oriented Programming 12 (4) (1999).
[29] X. Chen, B. Plimmer, CodeAnnotator: digital ink annotation within eclipse, in: Proceedings of the OZCHI, ACM, New York, 2007, pp. 211–214.
[30] B. Plimmer, J. Grundy, J. Hosking, R. Priest, Inking in the IDE: experiences with pen-based design and annotation, in: Proceedings of the VL/HCC, IEEE Press, New York, 2006, pp. 111–115.
[31] R. Priest, B. Plimmer, RCA: experiences with an IDE annotation tool, in: Proceedings of the CHINZ, ACM, New York, 2006, pp. 53–60.
[32] B. Plimmer, J. Grundy, Beautifying sketching-based design tool content: issues and experiences, in: Proceedings of the AUIC, Australian Comp. Soc., 2005, pp. 31–38.
[33] L. Yeung, B. Plimmer, B. Lobb, D. Elliffe, Levels of formality in diagram presentation, in: Proceedings of the OZCHI, ACM Press, New York, 2007, pp. 311–317.
[34] G. Taubin, Geometric signal processing on polygonal meshes, in: EUROGRAPHICS STAR Reports, 2000.
[35] V. Goel, Sketches for Thought, MIT Press, Cambridge, MA, 1995.
[36] B. Plimmer, M. Apperley, Interacting with sketched interface designs: an evaluation study, in: Proceedings of the CHI, ACM, New York, 2004, pp. 1337–1340.
[37] ITEA, The Trust4All project, 2005 ⟨www.win.tue.nl/trust4all⟩.
[38] A. Telea, AOI user study, 2007 ⟨www.win.tue.nl/∼alext/ARCHIVIEW/experiment.html⟩.
[39] R.A. Likert, A technique for the measurement of attitudes, Archives of Psychology 140 (1932).
[40] J. Sethian, Level Set Methods and Fast Marching Methods, Cambridge University Press, Cambridge, 1999.
[41] D. Reniers, A. Telea, Tolerance-based feature transforms, in: Advances in Computer Graphics and Computer Vision, vol. 4, Springer, Berlin, 2008, pp. 187–200.
[42] L.F. Costa, R.M. Cesar, Shape Analysis and Classification: Theory and Practice, CRC Press, Boca Raton, 2001.

[43] S. Berretti, A.D. Bimbo, P. Pala, Retrieval by shape similarity with perceptual distance and effective indexing, IEEE Transactions of Multimedia 2 (4) (2000) 225–239.
[44] D.M. Gavrila, Multi-feature hierarchical template matching using distance transforms, in: Proceedings of the ICPR, 1998, pp. 439–444.
[45] K. Gtauman, T. Darrell, Fast contour matching using approximate earth mover's distance, in: Proceedings of the CVPR, 2004, pp. 220–227.
[46] E. Bondarev, M. Chaudron, H. Byelas, P. de With, A toolkit for design and performance analysis of real-time component-based software systems, in: Proceedings of the International Conference on Software Engineering Advances (ICSEA), IEEE Press, New York, 2006, pp. 4–12.
[47] S. Arya, D. Mount, N. Netanyahu, R. Silverman, Y. Wu, An optimal algorithm for approximate nearest neighbor searching, Journal of the ACM 45 (1998) 891–923 ⟨www.cs.umd.edu/mount/ANN⟩.